DISTRIBUTING NOTIFICATIONS TO MULTIPLE RECIPIENTS VIA A BROADCAST LIST

TECHNICAL FIELD

[0001] Embodiments of the present invention relate to the field of software notifications. In particular, embodiments of this invention relate to distributing a notification or alert to multiple recipients via a broadcast list.

BACKGROUND OF THE INVENTION

[0002] Some prior systems broadcast messages to alert users to information (e.g., for example, news updates). The broadcast messages, alerts, or other notifications include individual packets of information sent to users. These systems broadcast the messages by sequentially sending individually addressed messages (e.g., multiple messages to one addressed user, or multiple instances of one message to several addressed users). That is, the alerts are sent one by one using multiple packets even though the same information is sent to multiple users. In such systems, an individual alert is sent for each recipient. Other systems only accommodate up to twenty recipients per message. However, these systems require the content provider to explicitly address each of the recipients.

[0003] Some systems use an electronic mail alias to distribute a single electronic mail message to multiple recipients. However, such a system is dependent on and limited to the electronic mail transport medium. There is a need for a system that transcends any one particular medium of delivery by using any of a plurality of transport mediums.

[0004] Accordingly, a system for distributing notifications to multiple recipients via a broadcast list is desired to address one or more of these and other disadvantages.

SUMMARY OF THE INVENTION

[0005] Embodiments of the invention include creating a broadcast list of recipients of a particular alert. Content providers send the broadcast alert to the broadcast list effecting delivery of the alert to each of the recipients on the broadcast list. In one embodiment, the broadcast list is managed and maintained by an alerts web service. A broadcast alert is intended for a scenario in which the same alert content is to be sent to multiple

recipients. In one embodiment, the broadcast list represents a set of users who want to receive content such as a daily news update, breaking news, a weather forecast, or a traffic report. With a broadcast list, a content provider only sends one alert to the broadcast list instead of sending thousands or millions of alerts with each one addressed to one person only.

[0006] The broadcast list of the invention provides scalability and allows alerts to be handled in a cost-effective way for both the content provider and distributor of alerts. Further, the broadcast list allows the content provider to offload the management and distribution of alerts to an alerts service and simplify the operations of the content provider. This reduces the packet traffic from the content providers to the alerts service significantly, and also allows the alerts service to scale better in terms of the number of packets processed per computing device associated with the web service and the reduction in database storage for the alert messages. Benefits of the broadcast list are significant particularly when the broadcast list size is fairly large (e.g., tens of thousands of members to several million members per list). Further, the invention transcends any one particular medium of delivery. That is, the invention may use multiple mediums such as electronic mail, instant messaging, and mobile short-message-service messaging.

[0007] In accordance with one aspect of the invention, a system processes a notification. The system includes a memory area to store a notification received from a third-party content provider. The notification includes routing information and content. The routing information includes a broadcast alias. The system also includes one or more computing devices to enable delivery of the stored notification to a plurality of users based on the broadcast alias. The system also includes a computer-readable medium to store computer-executable instructions to be executed on each of the computing devices to access the stored notification, determine a list of users associated with the broadcast alias for receiving the stored notification, and deliver the stored notification to the determined list of users.

[0008] In accordance with another aspect of the invention, a method processes a notification. The method includes receiving a data packet that represents a notification. The data packet includes routing information. The routing information includes a broadcast alias. The method also includes determining a list of recipients associated with

the broadcast alias to receive the data packet. The method also includes delivering the received data packet to the determined list of recipients.

[0009] In accordance with yet another aspect of the invention, one or more computer-readable media have computer-executable components for processing a notification. The components include an interface component for receiving a data packet representing a notification. The data packet includes routing information. The routing information includes a broadcast alias. The components include an address component for determining a list of recipients associated with the broadcast alias to receive the data packet. The components also include a broadcast component that delivers the received data packet to the determined list of recipients.

[0010] In accordance with still another aspect of the invention, a method accesses a memory area that stores a notification received from a third-party content provider. The notification includes routing information and content. The routing information includes a broadcast alias. The method also includes determining a list of recipients associated with the broadcast alias to receive the stored notification. The method also includes effecting the delivery of the received notification to each of the recipients on the determined list.

[0011] In accordance with another aspect of the invention, a web service processes a notification. The web service includes a memory area that stores a notification that is received from a third-party content provider. The notification includes routing information and content. The routing information includes a broadcast alias. The web service also includes one or more computing devices that enable delivery of the stored notification to a plurality of users based on the broadcast alias. The web service also includes computer-executable instructions that execute on each of the computing devices to access the stored notification, determine a list of users associated with the broadcast alias for receiving the stored notification, and deliver the stored notification to the determined list of users.

[0012] Alternatively, the invention may comprise various other methods and apparatuses.

[0013] Other features will be in part apparent and in part pointed out hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIG. 1 is a block diagram illustrating one example of a suitable alerts service environment in which the invention may be implemented.

[0015] FIG. 2 is an exemplary block diagram illustrating a detailed view of the alerts service illustrated in FIG. 1.

[0016] FIG. 3 is an exemplary flow chart illustrating operation of the alerts service.

[0017] FIG. 4A is an exemplary block diagram illustrating operation of software executed by a master router in the alerts service.

[0018] FIG. 4B is an exemplary block diagram illustrating operation of software executed by a worker router in the alerts service.

[0019] FIG. 5 is a block diagram illustrating one example of a suitable computing system environment in which the invention may be implemented.

[0020] Corresponding reference characters indicate corresponding parts throughout the drawings.

DETAILED DESCRIPTION OF THE INVENTION

[0021] In an embodiment, the invention processes the distribution of an alert or notification (e.g., event-driven content). In particular, the invention includes software for receiving an alert from a content provider and broadcasting the alert to a plurality of users. A broadcast alert is intended for a scenario in which the same alert content is to be sent to multiple recipients. In one embodiment, the broadcast list represents a set of users who want to receive content such as a daily news update, breaking news, a weather forecast, or a traffic report. In one embodiment, an alert carries time-sensitive content. Broadcasting an alert according to the invention allows the content provider to delegate explicit addressing of the alert to the alerts service. That is, the invention allows the content provider to transfer the addressing workload to the alerts service. The alerts service addresses the alert to each of the users associated with the broadcast alert, processes user preferences, and routes the addressed alerts to each of the intended users. Alternatively, the alerts service addresses the alerts but transfers the broadcasting workload to a gateway based on the user preferences (see FIG. 2 below). The gateway then performs the actual broadcasting of the alert to the intended users. In this

embodiment, the alerts service forwards a single alert with multiple explicitly addressed users to the gateway. Further, the invention includes thresholds for queue management to throttle requests incoming to the alerts service.

Alerts Environment

[0022] Referring first to FIG. 1, an exemplary block diagram illustrates one example of a suitable alerts service environment in which the invention may be implemented. In one example, the alerts service environment is referred to as a notification pipeline and database (NPD). FIG. 1 illustrates the communication flow between a content provider 102 such as content provider #1 through content provider #N, an alerts service 104, and a user device 106 such as user device #1 through user device #M. The content provider 102, the alerts service 104, and the user device 106 are coupled to a data communication network such as described with reference to FIG. 5 (see below). The content provider 102 sends an alert to the alerts service 104 for delivery to one or more of the user devices 106. The alerts service 104 accesses a subscription database 108 storing subscription information and user routing preferences 110 to determine which user device(s) 106 should receive the alert. The alerts service 104 then delivers the alert to the determined user device 106.

[0023] The user device 106 may be a computer such as computer 130 described with reference to FIG. 5. Further, the user device 106 may execute an alerts application (e.g., an instant messaging application) that receives and processes alerts. The alerts application executes on a user device 106 such as a cellular telephone (e.g., a Smartphone device), a pager, and a handheld computing device (e.g., a personal digital assistant or a Pocket PC device). Further, the user device 106 may include any of the above exemplary devices enabled with an information service such as a SMART PERSONAL OBJECTS TECHNOLOGY (SPOT) brand of telecommunication service and/or devices. The information service comprises a computing infrastructure (e.g., a telecommunication service) for sending data and information to personal and home devices via computer networks, wireless networks and the Internet. User devices 106 which may be enabled with the information service include, but are not limited to, the following devices: clocks, alarm clocks, radios incorporating clocks, watches, billfolds, wallets, checkbook

and passbook wallets, purses, pens, metal key rings, key holders, wireless devices, computer hardware (e.g., peripherals, monitors, and displays), electronic calendar devices, and refrigerator magnets. Further, magazines, books, and user manuals relating to computers, computer programs, personal information devices and wireless communications may also incorporate the information service. The information service enables billions of devices to communicate with each other. For example, customers select the type of information and services they want to receive on the enabled devices via a configuration web page. This content is subsequently beamed to and displayed on the device. Information available to users on devices using the information service includes personal messages, calendar updates, and customized news, weather, financial and sports information.

[0024] The alerts service illustrated in FIG. 1 transcends any one particular transport medium for delivery of notifications. The invention may use any of a plurality of transport mediums such as electronic mail, instant messaging, and mobile short-message-service messaging.

[0025] The system of FIG. 1 processes a notification or alert. The alerts service 102 includes a memory area storing a notification received from a third-party content provider 102. In one example, the memory area includes a plurality of databases. An interface component 112 receives a data packet representing the notification. The notification includes routing information and content. The routing information includes a broadcast alias. One or more computing devices associated with the alerts service 104 enable delivery of the stored notification to a plurality of users based on the broadcast alias. Software executing on each of the computing devices accesses the stored notification, determines a list of users associated with the broadcast alias for receiving the stored notification, and delivers the stored notification to the determined list of users. In one embodiment, an address component 114 determines the list of recipients associated with the broadcast alias to receive the notification. Further, a broadcast component 116 routes the notification to the determined list of recipients.

[0026] In one embodiment, the system of FIG. 1 is implemented as a web service. Further, functionality associated with the alerts service 104 of FIG. 1 may be distributed

among one or more computers. For example, the alerts service 104 may include a distributed processing system such as illustrated in FIG. 2.

[0027] Referring next to FIG. 2, an exemplary block diagram illustrates a detailed view of the alerts service 104 illustrated in FIG. 1. In this embodiment, the plurality of computing devices includes a master router 202 and a plurality of worker routers 204. The master router 202 associates the stored notification with one of the plurality of worker routers 204. Broadcast list processing is distributed by the master router 202 across the plurality of worker routers 204 such as NRouters. The worker routers 204 accept or reject the assigned workload based on their currently assigned workload. While any worker router 204 may act as a master router 202, the NRouter that originally receives the alert from the content provider 102 is designated as the master router 202. The master router 202 coordinates the processing of the received alert across other worker NRouters 204. The master router 202 also functions as a worker NRouter 204 to process alerts. The master NRouter 202 is responsible for logging the alert to a memory area such as an activity queue database 205. The master 202 and worker 204 routers communicate with each other through an STS (server-to-server) layer (see FIG. 4). The STS layer includes a communication component that provides a network connection between or among all routers in the system. The master router 202 and the worker routers 204 communicate with each other through an STS component. In one embodiment, a maximum of 256 worker NRouters 204 process a single broadcast list. However, there is no limit on the number of broadcast NRouters that may be deployed.

[0028] A unicast NRouter handles alert packets that are explicitly addressed to a particular user by the content provider 102. A broadcast NRouter or worker NRouter 204 processes alert packets that are addressed to a broadcast list. Both types of routers are built out of the same code base so that any NRouter may actually act as both unicast and broadcast at the same time if necessary to improve efficiency and scalability. All the broadcast NRouters are behind one fan-out logical store. The mappings are installed through an application program such as dbsmgmt. An NRouter is designated as a broadcast NRouter by adding the following entry into a configuration file such as soft.xml with appropriate values for the MAC and IP attributes and installing the mappings:

<server nam="nrouter_bcast" mac="..." ip="..." />

In another configuration file such as notifications.conf, a NPD_NROUTER_CLASS field is set to Broadcast in an [NPD] section.

[0029] The master 202 and worker 204 routers access a computer-readable medium storing a data structure (e.g., in a broadcast list database 206). The data structure may be associated with an application programming interface. The data structure includes a broadcast list identifier associated with the broadcast alias and a list of the users associated with the broadcast list identifier. In one embodiment, the broadcast list database 206 includes a broadcast list table 207 associating the broadcast list identifier with a specific member table 208 storing the list of users associated with the broadcast list identifier. The subscription database 108 or a user profile database stores the user routing preferences 110. Invention software routes the stored notification to the users on the determined list based on user routing preferences 110 corresponding thereto stored in the subscription database 108.

[0030] In another embodiment, the alerts service identifies the users associated with the broadcast alias, but sends the notification and the identified users to one or more third-party gateways for delivery. For example, the gateways may include an instant messaging gateway 210, a mobile gateway 212, and an electronic mail gateway 214. The third-party gateways route the notification to the user devices 106 associated with the identified users via one or more types of networks 216.

Alerts Service Operation

[0031] Referring next to FIG. 3, an exemplary flow chart illustrates operation of the alerts service. One or more computer-readable media have computer-executable instructions for performing the method illustrated in FIG. 3. The invention software or other computer-executable instructions receives a data packet representing a notification at 302. The data packet has routing information including a broadcast alias. The software populates an activity queue with the received data packet at 304 and subsequently accesses the activity queue to obtain the stored data packet. The invention software determines a list of recipients associated with the broadcast alias to receive the data packet at 306. In one embodiment, the software retrieves a broadcast list identifier

from a broadcast list table via the broadcast alias. The software identifies one or more recipients associated with the broadcast list identifier by accessing a member list table. The software delivers the received data packet to the determined list of recipients at 308.

[0032] In the master/worker router embodiment of FIG. 2, the software receives an alert from a content provider directed to a broadcast list alias. The master router parses the alert packet and verifies or validates the content provider. The master router stores the alert in the activity queue database. The master router accesses the broadcast list database using information in the packet such as the broadcast list alias. In one embodiment, the master router uses a content provider identifier and the alias to obtain the broadcast list identifier. There is a user/member table for each broadcast list identifier which lists all users/members associated with the broadcast list identifier. For example, the member table may be one table with multiple partitions. Every row in the member table has a broadcast list identifier, a sequence number (unique to the broadcast list identifier and user), and a unique identifier for each user.

[0033] The master router distributes processing based on the number of members for the broadcast list identifier and the number of worker routers available. The master router may queue multiple work items for each worker router. In one embodiment, the master router stores a table of the worker routers (including itself) in memory such as a state table. In another embodiment, the master router stores the table in a database such as a persistent SQL database. Storing the state information in a database provides reliability should the master router become unavailable.

[0034] Each worker router accepts assigned work (e.g., by accessing the activity queue) and processes the assigned work item by accessing the member table to obtain a list of users, loading user preferences (e.g., from a user profile database) for each user on the list, and delivering the alert to each user based on the preferences. In one embodiment, the user profile database includes a SQL table indexed by a user identifier, a user name, and routing preferences.

[0035] With the invention, each content provider sends a single packet to the alerts service for delivery to a broadcast list of users. In an alternative embodiment, the alerts service identifies the individual users associated with the broadcast list, and transfers the packet and the list of users to another computing device (e.g., a gateway) to perform the

routing. That is, the software sends the received data packet and the determined list of recipients to a third-party gateway for routing the received data packet to each of the recipients on the determined list. The software effects the routing of the received notification to each of the recipients on the determined list.

[0036] In particular, an NRouter identifies a packet addressed to a broadcast list by the presence of the attribute listed on the TO element as shown below:

<TO listid="..." >

In one form, the listid is in hexadecimal format and between the range 0 through (2^32) – 1(i.e., approximately four billion values). This corresponds to the npd_ListID column in the npd_BroadcastList table discussed below. The content provider provisions the listid with the alerts service.

[0037] Referring next to FIG. 4A, an exemplary block diagram illustrates operation of software executed by a master router in the alerts service. When a content provider posts an alert 402 such as in the form of an extensible markup language (XML) document to the master NRouter at 402, the master NRouter parses the alert 402 and validates the packet at 404. In one embodiment, the master NRouter applies usage and throttle limits. The master NRouter then acknowledges receipt of the alert 402 by returning a message with HSE_STATUS_PENDING status to the content provider. The master NRouter asynchronously processes the alert 402 by queuing the alert 402 into an NPD queue 406 or other memory area. The NPD queue 406 represents an internal queue of work items that are acted upon by any one of the multiple threads in the thread pool, but only one thread processes the item at any given time.

[0038] The master NRouter then retrieves information regarding the specific broadcast list specified in the alert 402, such as the number of members or users, and an internal 32-bit row identifier from a database such as BLdb. The master NRouter enforces the usage limits for the content provider, if any, and logs the alert 402 to an activity queue database such as AQdb 410. A copy of the broadcast activity is logged to every physical bucket in each activity queue physical store. This is done before returning a final status to the content provider. If the master NRouter is unsuccessful during any of the above processing, it returns an error code such as "500 Server Error" with status = 600. Otherwise, the master NRouter returns "202 Accepted" with status = 100.

[0039] A list processor module 412 associated with the master NRouter picks up the queued packet and uses the size of the broadcast list to determine an optimum number of processing jobs to schedule across the worker NRouters. These jobs are then queued into a scheduler queue 414 and handed off to a scheduler module 416 which packages each job into a request packet (e.g., as defined by a fan-out protocol) and assigns each job to an appropriate logical area of the logical NRouter in the STS layer 418. These logical areas map to the various NRouters. The scheduler module 416 also adds an entry into an outstanding jobs queue 420 with an appropriate expiration timestamp. The list row identifier, a tick-count and a chunk identifier together make up the key by which the outstanding jobs queue 420 is searched. A response command handler 422 or other monitor process executing on the master NRouter periodically checks the outstanding jobs queue 420 (e.g., every 5 minutes) and reschedules expired jobs if necessary. In one embodiment, the periodic interval is configurable and rescheduling occurs up to a total of three times before discarding the job.

[0040] Referring next to FIG. 4B, an exemplary block diagram illustrates operation of software executed by a worker router in the alerts service. On the worker NRouter, the request packet is picked up from the STS layer 424. If that worker NRouter is willing to accept this work item, a request command handler 426 sends an acknowledgement (ACK) back to the master NRouter. Otherwise, the request command handler 426 returns a not acknowledged (NACK) status (see the fan-out NRouter to NRouter protocol illustrated in FIG. 4A). One implementation returns the ack/nack immediately and does not wait for the worker NRouter to complete its task. The job is then queued into a fan-out request queue 428. A fan-out processor 430 reads the BLdb database 408 for a range of members defined by the chunk identifiers specified in the request packet. The fan-out processor 408 creates small blocks of members (e.g., 250 members per block) for further processing by a rules engine 432. These configurable blocks are added to an NPD queue 434. The policy threads in the rules engine 432 process each block independently.

[0041] The master NRouter maintains an in-memory table of its pending jobs. Since it is in-memory, the table is susceptible to master NRouter failures. To improve reliability, the master NRouter periodically writes out its state into a table such as a structured query language (SQL) table. The worker NRouters work off of the database to update the rows

corresponding to each work item belonging to a particular fan-out request. The master NRouter (or a secondary master) monitors the work items that are being updated in the database.

[0042] The following tables show exemplary request and response packet formats for use with the alerts service. The master NRouter sends the request packet to the worker NRouter to assign an alert to the worker NRouter. The worker NRouter accepts or declines the assigned alert via the response packet.

CMD	KEY	ChunkID Start	ChunkID End	XML Size	XML Packet

Table 1. Request Packet Format.

CMD	KEY	ChunkID Start	ChunkID End

Table 2. Response Packet Format.

[0043] The CMD field indicates one of the following: fan-out request processing (i.e., requests an NRouter to process a fan-out job), fan-out response accepted (i.e., worker NRouter accepts the task), or fan-out response declined (i.e., worker NRouter declines the task). The KEY field correlates outstanding jobs in the Master NRouter with the worker NRouter responses. In one example, the key field includes a List RowID and a Tick-count. The ChunkID Start and End fields specify an interval within the broadcast List member range that a worker NRouter is to process. In one example, the ChunkID start and end fields map to npd_Bucket in an npd_BroadcastListMembers table.

[0044] The fan-out processor may be fine-tuned using the following configuration settings (e.g., such as in a notifications.conf file). An NPD_BROADCAST_WORKER_IDEAL_JOB_SIZE setting represents an integral value that is used as a guideline by the master NRouter when assigning work to the worker NRouters. A default value includes 1000 list members per worker NRouter per request. An NPD_BROADCAST_JOB_EXPIRATION_TIMEOUT setting, in milliseconds, represents the time period to wait before attempting to reschedule the job with another worker NRouter. A default value includes one minute. An NPD_BROADCAST_MEMBERS_PER_POLICY_BLOCK setting represents the

maximum number of recipients that are associated with an instance of the policy block. A default value includes 250 recipients.

Broadcast List Databases

[0045] Exemplary databases and tables involved with broadcast list processing include npd_BLdb_rw_1 and npd_BLdb_ro_1. They both contain the tables npd_BroadcastList and npd_BroadcastListMembers, and both databases have identical content.

[0046] Web page front-end computers read and write to npd_BLdb_rw_1. The broadcast list provisioning also occurs on this database. The SQL instance hosting this database also acts as a distributor for transaction replication to populate npd_BLdb_ro_1. This latter database is used by the NRouter front ends at run time. The database npd_BLdb_rw_1 may be deployed on a different network switch than npd_BLdb_ro_1. These databases are accessed in NPD via the NSD_STORE_NPD_LISTS and NSD_STORE_NPD_LISTS_PROTECTED logical stores, respectively.

[0047] The following statements create these databases in the file soft.xml:

```
<?xml version="1.0"?>
<service n="..." >
  <physicalstore srv="hostName" dtyp="blrw" />
  <physicalstore srv="hostName" dtyp="blro" />
</service>
```

In the above statements, srv represents the appropriate host machine names.

[0048] Both BLdb RW and RO databases are setup to do Read-Only failover at physical bucket (PB) level. The number of PBs for the RW database is equal to the number of RO databases (physical stores), the number of PBs for the RO database is equal to 2 per physical store. The failover generation algorithm also takes into account the network switch if specified.

Npd_Broadcast_List Table

npd_CPID (4 Bytes)	npd_ListID (4 Bytes)	npd_RowID (4 Bytes)	npd_MemberCount (4 Bytes)	npd_Language (4 Bytes)	npd_DisplayName (64 Bytes)
CP1	ListID1	1	1000000	0	Breaking News
CP1	ListID2	2	50000	0	Local News
CP2	ListID1	3	15000	0	Seattle Traffic

Table 3. Exemplary Schema for npd_BroadcastList Table.

[0049] Each row in Npd_BroadcastList table represents one broadcast list. The npd_CPID and npd_ListID together uniquely identify a list, as does the npd_RowID column by itself. The npd_CPID column corresponds to the npd_Row column in the Content Provider table. This column contains the site ID of the content provider that owns the List. The npd_ListID column contains List owner (e.g., Content Provider) defined data, and its semantics are known to the NRouter. The owner has the entire range of values at its disposal. The npd_RowID column is the identity column that contains a unique 32-bit number, and is generally used within the NPD to identify Lists. This column is related to npd_RowID column in npd_BroadcastListMembers table. This column also corresponds to the npd_SubscriptionId column in the npd_Subscriptions table in the user profile database. In the case of broadcast activities in activity queue database, its value is used for npd_ReceiverIDRow column in npd_Activity2 table. In one example, the npd_RowID is a 32-bit number. The bits or a subset thereof may be organized or otherwise given specific meaning. For example, a subset of the bits may identify a specific alert such as an hourly news update or a nightly news update. In one embodiment, the invention software codes npd_CPID and npd_ListID to map to a specific npd_RowID for lookups and modification to a specific broadcast list.

[0050] The npd_MemberCount column contains the count of members belonging to each broadcast list. The master NRouter uses this count to decompose a fan-out request into an optimum number of jobs that are then distributed across worker NRouters. The npd_Language column holds locale information in one embodiment. The npd_DisplayName column contains a user-displayable string for the given broadcast list. There may be more than one row sharing the same topic information as long as the provider specific data portion is unique. The display name is available on the first row if

there are multiple rows, with the remaining rows containing a NULL for this column value.

Npd_Broadcast_List_Members Table

[0051] Each row in this exemplary table represents one subscription to a broadcast list.

npd_RowID (4 Bytes)	npd_Cluster (1 Byte)	npd_PUIDhigh (4 Bytes)	npd_PUIDlow (4 Bytes)
1	0	0x00000001	0x00000001
1	0	0x00000001	0x00000002
2	120	0x00000002	0x00000001
2	255	0x00000030	0x00000040
3	255	0x00000060	0x00000003
4	3	0x00000001	0x00000001
4	4	0x00000001	0x00000005
5	0	0x00000004	0x00000055
5	0	0x00000005	0x00000001
6	128	0x00000060	0x00000001
6	255	0x00000067	0x00000004

Table 4. SQL Schema for npd_BroadcastListMembers Table.

[0052] The npd_RowID column contains a 32-bit value that uniquely identifies a List. This column corresponds to the npd_RowID column in npd_BroadcastList table. The members of each List are grouped into one of the 256 possible npd_Cluster clusters. The master NRouter assigns one or more clusters at a time to each worker NRouter for processing. The value is based on a hash of the member PUID. The npd_PUIDhigh and npd_PUIDlow columns represent identifiers that uniquely identify a single user.

npd_Activity2 Table

[0053] The primary clustered index is based on npd_ReceiverIDtype, npd_ReceiverIDbucket, npd_ReceiverIDrow, and npd_ID. The npd_ReceiverIDtype ensures that all broadcast activities are grouped together and all unicast activities are

together so as to take advantage of SQL page caching, as each broadcast activity is read by more than one user. Typically, the read/write ratio for broadcast activities is very high compared to unicast activities.

[0054] The NRouter logs a copy of the broadcast activity to each of the physical buckets in the activity queue database. During a read operation, the user is directed to one of these copies based on a function of the user's bucket.

Activity Queue Server (AQS) .

[0055] The activity queue database stores the alerts for each user for a time interval (e.g., twenty-four hours) configurable by an administrator of the alerts service or by the user. In one example, a broadcast activity table includes multiple rows. Each row stores a user identifier and an alert. Each user may be allocated one or more rows. In another example, each message is stored once, while a list of recipients of the message is stored and associated with the message.

[0056] The AQS runs a separate set of threads to handle broadcast activity expiration enforcement, and dirty bucket cleanup due to failover. The related SQL stored procedures have been modified to take an extra argument to distinguish broadcast from unicast activities. Broadcast activities are cleaned up at the same frequency as unicast activities.

Broadcast List Programming Interface

[0057] Application programming interfaces (APIs) are exposed (e.g., as a web service) to content providers to create and maintain broadcast list IDs. In one example, this function is implemented by a subscription management service. Exemplary APIs for managing broadcast lists are described in Appendix A.

Exemplary Operating Environment

[0058] FIG. 5 shows one example of a general purpose computing device in the form of a computer 130. In one embodiment of the invention, a computer such as the computer 130 is suitable for use in the other figures illustrated and described herein. Computer 130 has one or more processors or processing units 132 and a system memory 134. In the

illustrated embodiment, a system bus 136 couples various system components including the system memory 134 to the processors 132. The bus 136 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0059] The computer 130 typically has at least some form of computer readable media. Computer readable media, which include both volatile and nonvolatile media, removable and non-removable media, may be any available medium that may be accessed by computer 130. By way of example and not limitation, computer readable media comprise computer storage media and communication media. Computer storage media include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. For example, computer storage media include RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium that may be used to store the desired information and that may be accessed by computer 130. Communication media typically embody computer readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and include any information delivery media. Those skilled in the art are familiar with the modulated data signal, which has one or more of its characteristics set or changed in such a manner as to encode information in the signal. Wired media, such as a wired network or direct-wired connection, and wireless media, such as acoustic, RF, infrared, and other wireless media, are examples of communication media. Combinations of the any of the above are also included within the scope of computer readable media.

[0060] The system memory 134 includes computer storage media in the form of removable and/or non-removable, volatile and/or nonvolatile memory. In the illustrated

embodiment, system memory 134 includes read only memory (ROM) 138 and random access memory (RAM) 140. A basic input/output system 142 (BIOS), containing the basic routines that help to transfer information between elements within computer 130, such as during start-up, is typically stored in ROM 138. RAM 140 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 132. By way of example, and not limitation, FIG. 5 illustrates operating system 144, application programs 146, other program modules 148, and program data 150.

[0061] The computer 130 may also include other removable/non-removable, volatile/nonvolatile computer storage media. For example, FIG. 5 illustrates a hard disk drive 154 that reads from or writes to non-removable, nonvolatile magnetic media. FIG. 5 also shows a magnetic disk drive 156 that reads from or writes to a removable, nonvolatile magnetic disk 158, and an optical disk drive 160 that reads from or writes to a removable, nonvolatile optical disk 162 such as a CD-ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that may be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 154, and magnetic disk drive 156 and optical disk drive 160 are typically connected to the system bus 136 by a non-volatile memory interface, such as interface 166.

[0062] The drives or other mass storage devices and their associated computer storage media discussed above and illustrated in FIG. 5, provide storage of computer readable instructions, data structures, program modules and other data for the computer 130. In FIG. 5, for example, hard disk drive 154 is illustrated as storing operating system 170, application programs 172, other program modules 174, and program data 176. Note that these components may either be the same as or different from operating system 144, application programs 146, other program modules 148, and program data 150. Operating system 170, application programs 172, other program modules 174, and program data 176 are given different numbers here to illustrate that, at a minimum, they are different copies.

[0063] A user may enter commands and information into computer 130 through input devices or user interface selection devices such as a keyboard 180 and a pointing device 182 (e.g., a mouse, trackball, pen, or touch pad). Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to processing unit 132 through a user input interface 184 that is coupled to system bus 136, but may be connected by other interface and bus structures, such as a parallel port, game port, or a Universal Serial Bus (USB). A monitor 188 or other type of display device is also connected to system bus 136 via an interface, such as a video interface 190. In addition to the monitor 188, computers often include other peripheral output devices (not shown) such as a printer and speakers, which may be connected through an output peripheral interface (not shown).

[0064] The computer 130 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 194. The remote computer 194 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 130. The logical connections depicted in FIG. 5 include a local area network (LAN) 196 and a wide area network (WAN) 198, but may also include other networks. LAN 136 and/or WAN 138 may be a wired network, a wireless network, a combination thereof, and so on. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and global computer networks (e.g., the Internet).

[0065] When used in a local area networking environment, computer 130 is connected to the LAN 196 through a network interface or adapter 186. When used in a wide area networking environment, computer 130 typically includes a modem 178 or other means for establishing communications over the WAN 198, such as the Internet. The modem 178, which may be internal or external, is connected to system bus 136 via the user input interface 184, or other appropriate mechanism. In a networked environment, program modules depicted relative to computer 130, or portions thereof, may be stored in a remote memory storage device (not shown). By way of example, and not limitation, FIG. 5 illustrates remote application programs 192 as residing on the memory device. The

network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0066] Generally, the data processors of computer 130 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the steps described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described herein.

[0067] For purposes of illustration, programs and other executable program components, such as the operating system, are illustrated herein as discrete blocks. It is recognized, however, that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

[0068] Although described in connection with an exemplary computing system environment, including computer 130, the invention is operational with numerous other general purpose or special purpose computing system environments or configurations. The computing system environment is not intended to suggest any limitation as to the scope of use or functionality of the invention. Moreover, the computing system environment should not be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, mobile telephones, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0069] The invention may be described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include, but are not limited to, routines, programs, objects, components, and data structures that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0070] An interface in the context of a software architecture includes a software module, component, code portion, or other sequence of computer-executable instructions. The interface includes, for example, a first module accessing a second module to perform computing tasks on behalf of the first module. The first and second modules include, in one example, application programming interfaces (APIs) such as provided by operating systems, component object model (COM) interfaces (e.g., for peer-to-peer application communication), and extensible markup language metadata interchange format (XMI) interfaces (e.g., for communication between web services).

[0071] The interface may be a tightly coupled, synchronous implementation such as in Java 2 Platform Enterprise Edition (J2EE), COM, or distributed COM (DCOM) examples. Alternatively or in addition, the interface may be a loosely coupled, asynchronous implementation such as in a web service (e.g., using the simple object access protocol). In general, the interface includes any combination of the following characteristics: tightly coupled, loosely coupled, synchronous, and asynchronous. Further, the interface may conform to a standard protocol, a proprietary protocol, or any combination of standard and proprietary protocols.

[0072] The interfaces described herein may all be part of a single interface or may be implemented as separate interfaces or any combination therein. The interfaces may execute locally or remotely to provide functionality. Further, the interfaces may include additional or less functionality than illustrated or described herein.

[0073] In operation, computer 130 executes computer-executable instructions such as those illustrated in FIG. 3 to process a notification by receiving the notification addressed

to a broadcast alias, determining the list of recipients associated with the broadcast alias, and delivering the notification to each recipient on the list.

[0074] The order of execution or performance of the methods illustrated and described herein is not essential, unless otherwise specified. That is, elements of the methods may be performed in any order, unless otherwise specified, and that the methods may include more or less elements than those disclosed herein.

[0075] When introducing elements of the present invention or the embodiment(s) thereof, the articles "a," "an," "the," and "said" are intended to mean that there are one or more of the elements. The terms "comprising," "including," and "having" are intended to be inclusive and mean that there may be additional elements other than the listed elements.

[0076] In view of the above, it will be seen that the several objects of the invention are achieved and other advantageous results attained.

[0077] As various changes could be made in the above constructions, products, and methods without departing from the scope of the invention, it is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

APPENDIX A

[0078] The following exemplary APIs enable management and maintenance of broadcast lists.

Broadcast List Table Related

[0079] Structure:

```
npdBroadcastList        bufBL;
```

[0080] To create a broadcast list:

Set

```
bufBL.ulCPID
bufBL.ulListID
bufBL.ulLanguage
bufBL.szDisplayName
```

Call

```
npdCreate(NSD_OBJECT_( NSD_STORE_NPD_LISTS,
NPDTYPE_LIST), &bufBL);
```

[0081] To get all the properties of a broadcast list:

Set

```
bufBL.ulCPID
bufBL.ulListID
```

Call

```
npdGet(NSD_OBJECT_( NSD_STORE_NPD_LISTS,
NPDTYPE_LIST), 0, &bufBL);
```

[0082] To iterate all the broadcast lists:

Set

```
bufBL.szDisplayName to non-empty string.
```

bufBL.ulLanguage to ~0 (to retrieve all language rows)

Call

npdGetIteratorHandle (NSD_OBJECT_(NSD_STORE_NPD_LISTS, NPDTYPE_LIST), NPDHOW_BYNAME, 0, &bufBL, ...);

Similarly for npdGetIterator.

Broadcast List Members Table Related

[0083] Structure:

npdBroadcastListMember bufBLM;

[0084] To add a member to a broadcast list:

Set

bufBLM.ulCPID

bufBLM.ulListID

bufBLM.ulPUIDhigh

bufBLM.ulPUIDlow

Call

npdCreate(NSD_OBJECT_(NSD_STORE_NPD_LISTS, NPDTYPE_LIST_MEMBER), &bufBLM);

[0085] To remove a member from a broadcast list:

Set

bufBLM.ulCPID

bufBLM.ulListID

bufBLM.ulPUIDhigh

bufBLM.ulPUIDlow

Call

npdDelete(NSD_OBJECT_(NSD_STORE_NPD_LISTS,
NPDTYPE_LIST_MEMBER), 0, &bufBLM);

[0086] To modify a broadcast list:

Use a combination of npdDelete and npdCreate to accomplish this operation.

[0087] To iterate all the broadcast list members:

Set

bufBLM.ulRowID – obtained from npd_BroadcastList table.

bufBLM.bCluster – Start of the range

bufBLM.bCluster2 – End of the range

bCluster = 0, bCluster2 = 255 returns the entire set of members, for large lists a subset is specified, bCluster = 0, bCluster2 = 1 to return approximately the first 20% of the list, bCluster = 2, bCluster2 = 3 the next 20% and so on.

Call

npdGetIteratorHandle (NSD_OBJECT_(
NSD_STORE_NPD_LISTS_PROTECTED , NPDTYPE_LIST),
NPDHOW_BYID, &bufBLM, ...);

Similarly for npdGetIterator. The store type specified is the BLdb_ro, but BLdb_rw is also appropriate depending on the need.

Subscriptions Table Related

[0088] Structure:

npdListSubscription bufLS;

[0089] To create a broadcast list subscription for a user:

Set

bufLS.ulCPID

bufLS.ulListID

bufLS.ulUserPUIDhigh

bufLS.ulUserPUIDlow

bufLS.bufSubscription.ulRouting

Call

npdCreate(NSD_OBJECT_(NSD_STORE_NPD_USER,
NPDTYPE_SUBSCRIPTION_LIST), &bufLS);

This does two things. 1) Creates an entry in the broadcast list member table 2) Creates an entry in the user subscriptions table.

[0090] To delete a single broadcast list subscription of a user:

Set

bufLS.ulCPID

bufLS.ulListID

bufLS.ulUserPUIDhigh

bufLS.ulUserPUIDlow

Call

npdDelete(NSD_OBJECT_(NSD_STORE_NPD_USER,
NPDTYPE_SUBSCRIPTION_LIST), NPDHOW_BYLISTS, &bufLS);

This does two things. 1) Deletes the entry in the broadcast list member table 2) Deletes the entry in the user subscriptions table.

[0091] To delete all broadcast list subscriptions for a content provider of a user:

Set

bufLS.ulCPID

bufLS.ulUserPUIDhigh

bufLS.ulUserPUIDlow

Call

npdDelete(NSD_OBJECT_(NSD_STORE_NPD_USER, NPDTYPE_SUBSCRIPTION_LIST), NPDHOW_BYCP, &bufLS);

This does two things. 1) Deletes all entries that belong to the given content provider for the user in the broadcast list member table 2) Deletes all the list subscription entries in the user subscriptions table that belong to the given content provider.

Activity Table Related

[0092] Structure:

npdBroadcastActivity bufBA (same as npdActivity)

[0093] To create a broadcast activity:

To create a copy of the broadcast activity in every physical bucket of each AQ logical store, use npdModify instead of npdCreate.

Set

All the appropriate fields in bufBA, similar to npdActivity except the following:

ulReceiverIDrow - contains the value of ulRowID column in npdBroadcastList table, for the given list.

wReceiverIDbucket – is automatically computed by the API.

usReceiverIDtype – should be NPDTYPE_LIST

Call

npdModify(NSD_OBJECT_(
NSD_STORE_NPD_BROADCASTACTIVITY,
NPDTYPE_BROADCASTACTIVITY),
NPDHOW_LOG_BROADCAST, &bufBA);

[0094] To get all broadcast activities belonging to a user:

Since the activities are filed under the list row identifier, the list row identifiers may be found by determining all the list subscriptions of the user. This information is input into a special query to gather all broadcast activities for the given user.

Set

bufBA.ulReceiverIDrow – User's rowID.
bufBA. wReceiverIDbucket – User's bucket

Call

npdGetIteratorHandle (NSD_OBJECT_(
NSD_STORE_NPD_BROADCASTACTIVITY,
NPDTYPE_BROADCASTACTIVITY), NPDHOW_BYLISTS,
&bufBA, ...);

Similarly for npdGetIterator.